Introduction à la cryptographie
TD6 – RSA : exemples simples et signatures

Cécile Pierrot

14 novembre 2019

L'objectif de ce TD est d'abord de voir quelques exemples simplifiés et concrets d'utilisation de RSA. Ensuite, nous allons voir à quel point la cryptographie est un parcours semé d'embûches, y compris lors de l'implémentation d'une simple primitive. Nous utiliserons pour cela l'exemple de l'algorithme de signature asymétrique RSA, telle qu'il est par exemple utilisé dans les cartes bancaires.

1 Prérequis et premiers exemples

1.1 Quelques rappels : définitions et propriétés du modulo

Soit n un entier naturel. L'ensemble $\mathbb{Z}/n\mathbb{Z}$ est défini comme l'ensemble des entiers modulo n, c'est-à-dire $\{0, 1, 2, \cdots, n-2, n-1\}$. Un entier naturel est dit premier s'il existe exactement deux entiers naturels qui le divisent : 1 et lui-même. Par exemple $3, 7, 43$ sont des nombres premiers mais $0, 1, 8$ et 15 ne le sont pas. Attention à ne pas confondre avec la définition de deux nombres premiers *entre-eux* : m et n sont premiers entre-eux si et seulement si leur seul diviseur commun est 1. Cela signifie que tout nombre premier est premier avec n'importe quel autre entier, mais deux nombres peuvent être premiers entre-eux, sans être premiers. Par exemple $4 et 21$ sont premiers entre-eux.
Soient a et b deux entiers relatifs. Alors on écrit $a \equiv b \pmod{n}$ et on dit que a est congru à b modulo n si et seulement si n divise $|a - b|$. Cela est équivalent à dire qu'il existe q un entier naturel tel que $a = n \cdot q + b$. En particulier, si r est le reste de la division euclidienne de a par n alors $a \equiv r \pmod{n}$. Il existe de nombreuses manières de noter $a \equiv b \pmod{n}$ et elles sont toutes équivalentes. On peut par exemple aussi écrire :

— $a = b \mod n$
— $a \equiv b \mod n$
— $a \equiv b \quad [n]$
— $a = b \quad [n]$.

Question 1. Vérifiez que $187 \equiv 2 \mod 5$, que $80 \equiv 0 \mod 8$ et que $-7 \equiv 6 \mod 13$.

Quelques propriétés à avoir en tête, que nous ne montrerons pas. Soient, a, b, des entiers relatifs et k et n des entiers naturels. Alors :

1. $n^k \equiv 0 \mod n$
2. Si a est premier avec n alors il existe un entier que l'on note a^{-1} et que l'on appelle l'inverse modulaire de a tel que $a \cdot a^{-1} \equiv 1 \mod n$.
3. Si $a \equiv b \mod n$ alors $a \cdot k \equiv b \cdot k \mod n$. Attention la réciproque est fausse.
4. Si $a \cdot k \equiv b \cdot k \mod n$ et que k divise n alors $a \equiv b \mod n/k$

Question 2. Qui est l'inverse modulaire de 3 modulo 7 ? 3 a-t-il une inverse modulaire modulo 18 ?

Question 3. Montrez l'item 3 et l'item 4.

1.2 Pourquoi RSA fonctionne-t-il ?

Lors du déchiffrement, on calcule le chiffré $c \equiv m^e \mod N$ à la puissance l'entier public d, le tout modulo N. Nous rappelons que :

$$e \cdot d \equiv 1 \mod (p-1)(q-1).$$

Pour garantir que l'on retrouve bien le message, nous avons besoin du théorème suivant, que l'on ne prouvera pas :

Théorème 1 (Petit théorème de Fermat)**.** Soient p un entier premier et a un entier naturel premier avec p. Alors :

$$a^{p-1} \equiv 1 \mod p.$$

Question 4. Montrez qu'en calculant $c^d \mod N$ on obtient bien le message initial $m \mod N$. Indice : appliquez deux fois le théorème de Fermat sur un entier a bien choisi, une fois modulo p, et une fois modulo q.

1.3 Comment calculer l'exposant secret d ?

Supposons que e, p et q soient connus. Pour le moment nous faisons l'hypothèse qu'il existe bien une inverse modulaire pour e, pour le modulo $(p-1)(q-1)$. Mais ceci n'est rien d'évident, et il faut en particulier vérifier certains conditions.
Supposons donc que d existe. Pour calculer cet exposant d tel que $e \cdot d \equiv 1 \mod (p-1)(q-1)$ il s'agit de savoir retrouver l'inverse modulaire d'un entier modulo $(p-1)(q-1)$. On pourrait procéder par recherche exhaustive, en testant tous les entiers modulos $(p-1)(q-1)$ mais cela serait beaucoup trop long. La bonne idée consiste à appliquer l'algorithme d'Euclide étendu.
L'algorithme d'Euclide étendu est une version de l'algorithme d'Euclide (qui pour rappel, permet de calculer le pgcd de deux entiers) dans laquelle, à partir de deux entiers a et b pris en entrée, nous calculons non seulement le pgcd de a et b, mais aussi la valeur de deux autres entiers relatifs u et v tels que :

$$a \cdot u + b \cdot v = \text{pgcd}(a, b).$$

Question 5. Que vaut le pgcd de deux entiers a et b si a et b sont premiers entre eux ? Dans ce cas, si l'on suppose que e est premier avec $(p-1)(q-1)$, quelles sont les valeurs qu'il faut prendre en entrée de cet algorithme pour obtenir l'inverse modulaire de e ?

Question 6. L'algorithme est donné en pseudo-code dans la figure ci-dessous. Appliquez à la main pour $a = 3$ et $b = 7$. Qui est alors l'inverse modulaire de 3 modulo 7 ?

```
Entrée : a, b entiers (naturels)
Sortie : r entier (naturel) et  u, v entiers relatifs tels que r = pgcd(a, b) et r = a*u+b*v

Initialisation : (r, u, v, r', u', v') := (a, 1, 0, b, 0, 1)
                 q  quotient entier

les égalités r = a*u+b*v et r' = a*u'+b*v' sont des invariants de boucle

tant que (r' ≠ 0) faire
    q := r÷r'
    (r, u, v, r', u', v') := (r', u', v', r - q *r', u - q*u', v - q*v')
    fait
renvoyer (r, u, v)
```

1.4 Un exemple simple

Maintenant que vous êtes armés, nous allons (enfin !) calculer le chiffrement d'un message (super simple). Prenons $p = 7$ et $q = 13$. Choisissons $e = 5$.

Question 7. Qui est la clef secrète ? La clef publique ?

```
def bezout(a, b):
    r, u, v, rr, uu, vv = (a, 1, 0, b, 0, 1)
    while rr:
        q = r // rr
        r, u, v, rr, uu, vv = (rr, uu, vv, r - (q*rr), u - (q*uu), v - (q*vv))
    return r, u, v
```

Supposons maintenant que le message clair soit $m = 18$.

Question 8. Quel est le message chiffré correspondant ?

Question 9. Et si l'on dispose d'un message chiffré $c \equiv 6 \mod 91$. Quel est le message clair ? Même question avec $c \equiv 1 \mod 91$.

2 La signature RSA

2.1 Quelques rappels

Pour rappel, étant donné un paramètre de sécurité n, la génération d'une paire de clés publique et privée pour RSA s'effectue de la manière suivante :

1. choisir au hasard deux nombres premiers distincts p et q de $n/2$ bits chacun ;
2. calculer le *module* $N = p \times q$, ainsi que l'indicatrice d'Euler $\varphi(N) = (p-1)(q-1)$;
3. choisir un *exposant public* $e \in \mathbb{Z}/\varphi(N)\mathbb{Z}$, non nul et co-premier avec $\varphi(N)$;
4. calculer l'*exposant privé* $d \in \mathbb{Z}/\varphi(N)\mathbb{Z}$ tel que $ed \equiv 1 \pmod{\varphi(N)}$ (un tel d existe toujours et il est unique ; on le calcule avec un algorithme d'*inversion modulaire* comme l'algorithme d'Euclide étendu) ;
5. la *clé publique* est alors le couple $PK = (e, N)$, et la *clé privée* $SK = (d, N)$.

Telles que les clés publique et privée ont été calculées, on a

$$M^{ed} \bmod N = M \text{ pour tout message } M \in \mathbb{Z}/N\mathbb{Z}.$$

Question 1. Quelle est la taille (en bits) du module N ? Et celle de $\varphi(N)$? Actuellement, quelles valeurs sont recommandées pour n ?

Question 2. Expliquez en quoi ce n'est pas une bonne idée de prendre $e = 1$.

Pour signer un message $M \in \mathbb{Z}/N\mathbb{Z}$ avec la clé privée (d, N), on calcule la signature $S = \mathrm{Sig}_{SK}(M) = M^d \bmod N$.

Question 3. Grâce à la clé publique, comment peut-on vérifier qu'une signature S correspond bien à un message M donné ?

2.2 Un petit exemple

Prenons par exemple $p = 5$, $q = 11$, et $e = 3$.

Question 4. Combien valent N, $\varphi(N)$, et d ?

Considérons alors le message $M = 13$ et les signatures $S_1 = 9$ et $S_2 = 7$.

Question 5. Laquelle de ces deux signatures correspond à M ?

Question 6. Calculez (à la main) une signature valide pour le message $M = 7$. Fastidieux, n'est-il pas ? Essayez de ruser pour ne pas avoir à calculer 26 produits modulo N.

3 Calcul d'exponentiation modulaire

3.1 Algorithme naïf

Comme vous avez pu vous en rendre compte à la question précédente, le calcul de $M^d \bmod N$ (on appelle ça une *exponentiation modulaire*) coûte cher, surtout s'il est effectué naïvement. Mais cher comment ?

Question 1. En admettant que, par construction, l'exposant secret d prend une valeur quelconque dans $\mathbb{Z}/\varphi(N)\mathbb{Z}$, quelle est sa taille (en bits) ?

Question 2. Quelle est la taille (en bits) de l'entier M^d ? Pour les valeurs recommandées de n, est-ce que cela est envisageable ? Comment remédier à ce problème ?

Question 3. Combien de multiplications modulaires sont-elles nécessaires pour calculer une signature $S = M^d \bmod N$? Encore une fois, pour les valeurs recommandées de n, est-ce que cela est envisageable ?

3.2 Algorithme d'exponentiation binaire

Nous allons tenter ici de voir comment faire mieux qu'un algorithme en temps linéaire en d (c'est-à-dire exponentiel en n) pour calculer $M^d \bmod N$.

Question 4. Montrez que, si d est pair (c'est-à-dire $d = 2d'$), alors vous pouvez ramener le calcul de $M^d \bmod N$ au calcul de $M^{d'} \bmod N$ suivi d'un carré modulo N.

Question 5. Que faire lorsque d est impair (c'est-à-dire $d = 2d' + 1$) ?

Question 6. Si d fait n bits, quelle taille fait d' dans les deux questions précédentes ?

Question 7. On peut alors répéter l'opération pour le calcul de $M^{d'} \bmod N$, et ainsi de suite. Déduisez-en un algorithme en temps $O(n)$ pour calculer $M^d \bmod N$. Quel est son coût ?
Astuce : Considérez $d = (d_{n-1} \dots d_1 d_0)_2 = \sum_{i=0}^{n-1} d_i 2^i$ l'écriture en base 2 de l'exposant d.

Question 8. Utilisez cet algorithme afin de calculer $9^{17} \bmod 55$. De combien de carrés et de multiplications modulo 55 avez-vous eu besoin ?

4 Attaques par canaux cachés

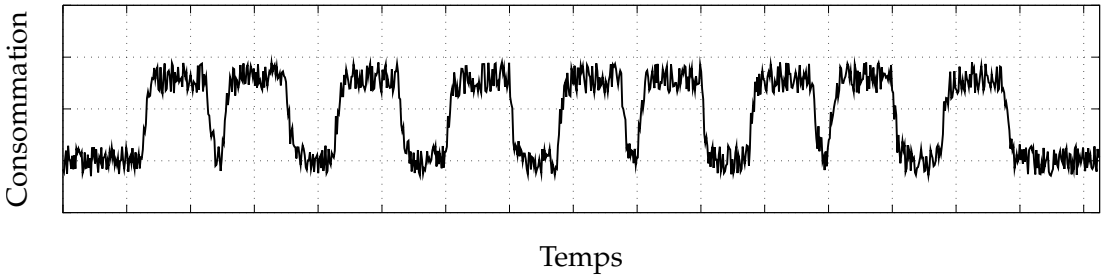
Nous nous plaçons ici dans le cadre de l'algorithme de signature RSA tel qu'il peut être utilisé par une carte bancaire, par exemple lors de l'authentification dynamique du protocole EMV. Dans ce contexte, l'alimentation électrique de la puce de la carte bancaire provient du terminal dans lequel elle est insérée. De la même manière qu'il est possible que des *skimmers* analysent et modifient les données qui circulent entre le terminal et la carte, il est aussi possible de construire un *skimmer* qui mesure très précisément (voire modifie) le courant électrique alimentant la puce, à la manière d'un oscilloscope.
Nous allons voir comment, dans un tel contexte, une implémentation naïve de l'algorithme d'exponentiation binaire peut révéler l'exposant secret d de la carte.

4.1 Analyse simple de consommation

Question 1. Dans l'algorithme d'exponentiation binaire obtenu aux questions précédentes, constatez que les calculs effectués à chaque itération dépendent des bits de l'exposant secret d.

Il se trouve que les opérations modulo N sont coûteuses : les nombres manipulés font en effet tous n bits. Lors du calcul d'un carré ou d'un produit modulaire, la puce va ainsi consommer plus que durant le reste de l'exécution de son programme.

Question 2. Commentez la trace de consommation ci-dessous, mesurée lors du calcul d'une exponentiation binaire pour un exposant de $n = 6$ bits. Retrouvez la valeur de l'exposant secret d.



Question 3. Quelle contre-mesure simple peut-on mettre en œuvre au niveau de l'algorithme d'exponentiation afin d'éviter cette attaque ? Quel est son coût ?

4.2 Attaque par injection de fautes

En contrôlant la tension d'alimentation de la puce, il est possible de baisser celle-ci très ponctuellement afin de lui faire faire des erreurs de calcul à des moments précis de l'exécution de son programme. (Une forte perturbation électromagnétique peut aussi faire l'affaire.)

Question 4. Expliquez comment l'injection de fautes peut vous permettre de détecter des opérations inutiles dans l'exécution d'un algorithme. Déduisez-en un moyen de déjouer la contre-mesure précédente.

En 1987, Montgomery proposa l'algorithme suivant, appelé *échelle de Montgomery* (ou *Montgo-*

mery powering ladder) :

$T_0 \leftarrow 1$
$T_1 \leftarrow M$
for $i \leftarrow n-1$ **downto** 0 **do**
    $T_{1-d_i} \leftarrow T_0 \times T_1 \bmod N$
    $T_{d_i} \leftarrow T_{d_i}^2 \bmod N$
return T_0

Question 5. Quel est le coût de cet algorithme ?

Question 6. Montrez qu'au début de chaque itération de l'algorithme, on a toujours $T_1 = T_0 \times M \bmod N$.

Question 7. Montrez qu'après j itérations de l'algorithme ($j \leq n$), on a bien $T_0 = M^{d^{(j)}} \bmod N$, avec $d^{(j)} = (d_{n-1} \dots d_{n-j})_2 = \lfloor d/2^{n-j} \rfloor$, l'entier formé par les j bits de poids forts de d.

Question 8. Que calcule l'échelle de Montgomery ?

Question 9. Montrez que cet algorithme n'est pas vulnérable aux injections de fautes.